AEROSPACE ENGINEERING

Course: Gas Dynamics

Course Instructor: Dr. Mohammed Madbouly

Name: Ghada Abdullah Ahmed Hassan

Section: 2

Report: Numerical Analysis of 1D Compressible Flow in a Convergent-Divergent Nozzle with Heat Transfer and Friction

1. Introduction

This report presents a numerical investigation of steady, one-dimensional compressible flow in a convergent-divergent nozzle, accounting for the combined effects of area change, wall friction, and heat transfer. The analysis is based on the governing equations of gas dynamics and employs a forward Euler integration scheme to solve the differential equations along the nozzle length.

The study compares four distinct cases:

- **Case 1**: Isentropic flow (area change only)
- **Case 2**: Area change with heat addition
- **Case 3**: Area change with wall friction
- **Case 4**: Area change with both heat addition and friction

2. Theoretical Background

2.1 Governing Equations

For steady, one-dimensional compressible flow in a variable area duct with friction and heat transfer, the governing equations can be expressed in differential form:

from continuity equation

$$\frac{d\rho}{\rho}+\frac{dV}{V}+\frac{dA}{A}=0 \quad (1)$$

from momentum equation

$$\frac{dp}{p}+\gamma M^2\frac{dV}{V}+\frac{\gamma}{2}M^2\left[\frac{fdx}{D}\right]=0 \quad (2)$$

from energy equation

$$\frac{dT}{T}+(\gamma-1)M^2\frac{dV}{V}=\chi\frac{dq}{C_pT_o} \quad (3)$$

from state equation

$$\frac{dp}{p}=\frac{dT}{T}+\frac{d\rho}{\rho} \quad (4)$$

from Mach number definition $\rightarrow$

$$\frac{dM}{M}=\frac{dV}{V}-\frac{1}{2}\frac{dT}{T} \quad (5)$$

where

$$\chi=\left[1+\frac{(\gamma-1)}{2}M^2\right]$$

from equations (3) & (5)

$$\frac{dT}{T}=\left(\frac{dq}{C_pT_o}\right)-\left[\frac{(\gamma-1)M^2}{\chi}\right]\frac{dM}{M} \quad (6)$$

from equations (5) & (6)

$$\frac{dV}{V}=\frac{1}{2}\left(\frac{dq}{C_pT_o}\right)+\left[\frac{1}{\chi}\right]\frac{dM}{M} \quad (7)$$

from equations (1) & (7)

$$\frac{d\rho}{\rho}=-\frac{1}{2}\left(\frac{dq}{C_pT_o}\right)-\left[\frac{1}{\chi}\right]\frac{dM}{M}-\frac{dA}{A} \quad (8)$$

from equations (2) & (7)

$$\frac{dp}{p}=-\frac{\gamma M^2}{2}\left(\frac{dq}{C_pT_o}\right)-\left[\frac{\gamma M^2}{\chi}\right]\frac{dM}{M}-\frac{\gamma M^2}{2}\left[\frac{fdx}{D}\right] \quad (9)$$

substitute equations (6),(8), (9) into equation (4)

$$\frac{dM}{M}=\left[\frac{(1+\gamma M^2)\chi}{2(1-M^2)}\right]\left(\frac{dq}{C_pT_o}\right)+\left[\frac{\gamma M^2\chi}{2(1-M^2)}\right]\left(\frac{fdx}{D}\right)-\left[\frac{2+(\gamma-1)M^2}{2(1-M^2)}\right]\frac{dA}{A} \quad (10)$$

where

$$\chi=\left[1+\frac{(\gamma-1)}{2}M^2\right]$$

2.3 Nozzle Geometry

The convergent-divergent nozzle is defined using third-order polynomial equations to ensure smooth area variation. For the convergent section:

Convergent part equations

$$A = c_o + c_2 (x-x_1)^2 + c_3 (x-x_1)^3$$

$$dA/dx = 2\, c_2 (x-x_1) + 3\, c_3 (x-x_1)^2$$

$$\text{with } c_o = A_1 \;\&\; c_2 = 3 (A_2 - A_1) / (x_2 - x_1)^2 \;\&\; c_3 = -2 (A_2 - A_1) / (x_2 - x_1)^3$$

For the divergent section:

Divergent part equations

$$A = d_o + d_2 (x-x_2)^2 + d_3 (x-x_2)^3$$

$$dA/dx = 2\, d_2 (x-x_2) + 3\, d_3 (x-x_2)^2$$

$$\text{with } d_o = A_2 \;\&\; d_2 = 3 (A_3 - A_2) / (x_3 - x_2)^2 \;\&\; d_3 = -2 (A_3 - A_2) / (x_3 - x_2)^3$$

Geometric Parameters:

At the inlet section	$x_1 = 0$	$\& \; A_1 = 0.5612\ m^2$
At the throat section	$x_2 = 0.2113\ m$	$\& \; A_2 = 0.1403\ m^2$
At the exit section	$x_3 = 1.0\ m$	$\& \; A_3 = 0.5612\ m^2$

All plots (nozzle geometry, Mach number, static and total pressure, static and total temperature, velocity, static density, and mass flow rate distributions) are generated automatically during code execution and appear inline within the MATLAB Live Script output.

```
%% 1D Compressible Flow in a Convergent-Divergent Nozzle
% This program solves steady, one-dimensional compressible flow in a
% variable area duct with the combined effects of area change, friction,
% and heat transfer. The numerical solution is obtained by integrating the
% differential equations along the duct length.
%
% The nozzle geometry is defined by third-order polynomial equations for
% the convergent and divergent sections, with a specified area ratio.
%
% Four cases are considered:
%   1. Area change only (isentropic flow)
%   2. Area change with heat addition
%   3. Area change with friction
%   4. Area change with both heat addition and friction
%
% Author: Ghada Abdullah, based on Prof. Dr. Mohamed Madbouli Abdelrahman's course
% Numerical Solution of Gas Dynamics Problem

clear all; close all; clc;

%% ========================================================================
%  PHYSICAL CONSTANTS AND GAS PROPERTIES
% ========================================================================
gama = 1.4;                     % Specific heat ratio (for air)
Rgas = 287;                     % Specific gas constant for air [J/(kg·K)]
Cp = (gama * Rgas) / (gama - 1); % Specific heat at constant pressure [J/(kg·K)]

%% ========================================================================
%  INLET CONDITIONS (Given at the nozzle inlet)
% ========================================================================
% Inlet conditions from the problem statement
Ma_inlet = 0.137;               % Inlet Mach number
TT_inlet = 840;                 % Inlet total temperature [K]
PT_inlet = 3721.7e3;            % Inlet total pressure [Pa] (3721.7 kPa)

%% ========================================================================
%  OPERATING CONDITIONS
% ========================================================================
Q_total = 400e3;                % Total heat added from inlet to exit [J/kg]
friction_factor = 0.02;         % Darcy friction factor (dimensionless)
XL = 1.0;                       % Total nozzle length [m]

%% ========================================================================
%  NUMERICAL SETUP
% ========================================================================
imax = 101;                     % Number of grid points (101 points along the duct)
dx = XL / (imax - 1);           % Step size [m]
dQ = Q_total * dx / XL;         % Heat added per step [J/kg]

% Pre-calculate the heat addition per step in dimensionless form
dQ_over_Cp_TT = dQ / (Cp * TT_inlet);  % Dimensionless heat addition
```

```
%% ========================================================================
%  NOZZLE GEOMETRY DEFINITION
% ========================================================================
% Initialize geometry arrays
x_coordinate = zeros(1, imax);      % x-coordinate along nozzle [m]
area = zeros(1, imax);              % Cross-sectional area at each point [m²]
diameter = zeros(1, imax);          % Diameter at each point [m]
dA_dx = zeros(1, imax);             % Derivative of area with respect to x

% Define the nozzle using third-order polynomials (convergent-divergent)
% Given data:
%   Inlet:  x1 = 0,      A1 = 0.5612 m²
%   Throat: x2 = 0.2113, A2 = 0.1403 m²
%   Exit:   x3 = 1.0,    A3 = 0.5612 m²

x1 = 0;         A1 = 0.5612;        % Inlet section
x2 = 0.2113;    A2 = 0.1403;        % Throat section
x3 = 1.0;       A3 = 0.5612;        % Exit section

% Coefficients for the convergent part (from x1 to x2)
c0 = A1;
c2 = 3 * (A2 - A1) / (x2 - x1)^2;
c3 = -2 * (A2 - A1) / (x2 - x1)^3;

% Coefficients for the divergent part (from x2 to x3)
d0 = A2;
d2 = 3 * (A3 - A2) / (x3 - x2)^2;
d3 = -2 * (A3 - A2) / (x3 - x2)^3;

% Generate the geometry along the nozzle length
for i = 1:imax
    x_coordinate(i) = (i - 1) * dx;     % Current x-position [m]

    % Determine which part of the nozzle we are in
    if x_coordinate(i) <= x2
        % Convergent section (inlet to throat)
        x_local = x_coordinate(i) - x1;
        area(i) = c0 + c2 * x_local^2 + c3 * x_local^3;
        dA_dx(i) = 2 * c2 * x_local + 3 * c3 * x_local^2;
    else
        % Divergent section (throat to exit)
        x_local = x_coordinate(i) - x2;
        area(i) = d0 + d2 * x_local^2 + d3 * x_local^3;
        dA_dx(i) = 2 * d2 * x_local + 3 * d3 * x_local^2;
    end

    % Calculate diameter from cross-sectional area (assuming circular cross-section)
    diameter(i) = sqrt(4 * area(i) / pi);
end

%% ========================================================================
%  INITIALIZE STORAGE ARRAYS FOR ALL FOUR CASES
```

```
% ========================================================================
% Case 1: Area change only (isentropic, no friction, no heat)
case1_Ma = zeros(1, imax);      % Mach number
case1_Velocity = zeros(1, imax);% Velocity [m/s]
case1_T_static = zeros(1, imax);% Static temperature [K]
case1_P_static = zeros(1, imax);% Static pressure [Pa]
case1_Rho_static = zeros(1, imax);% Static density [kg/m³]
case1_SoundSpeed = zeros(1, imax);% Speed of sound [m/s]
case1_T_total = zeros(1, imax); % Total temperature [K]
case1_P_total = zeros(1, imax); % Total pressure [Pa]
case1_Rho_total = zeros(1, imax);% Total density [kg/m³]
case1_massflow = zeros(1, imax);% Mass flow rate [kg/s]

% Case 2: Area change with heat addition only
case2_Ma = zeros(1, imax);
case2_Velocity = zeros(1, imax);
case2_T_static = zeros(1, imax);
case2_P_static = zeros(1, imax);
case2_Rho_static = zeros(1, imax);
case2_SoundSpeed = zeros(1, imax);
case2_T_total = zeros(1, imax);
case2_P_total = zeros(1, imax);
case2_Rho_total = zeros(1, imax);
case2_massflow = zeros(1, imax);

% Case 3: Area change with friction only
case3_Ma = zeros(1, imax);
case3_Velocity = zeros(1, imax);
case3_T_static = zeros(1, imax);
case3_P_static = zeros(1, imax);
case3_Rho_static = zeros(1, imax);
case3_SoundSpeed = zeros(1, imax);
case3_T_total = zeros(1, imax);
case3_P_total = zeros(1, imax);
case3_Rho_total = zeros(1, imax);
case3_massflow = zeros(1, imax);

% Case 4: Area change with both heat addition and friction
case4_Ma = zeros(1, imax);
case4_Velocity = zeros(1, imax);
case4_T_static = zeros(1, imax);
case4_P_static = zeros(1, imax);
case4_Rho_static = zeros(1, imax);
case4_SoundSpeed = zeros(1, imax);
case4_T_total = zeros(1, imax);
case4_P_total = zeros(1, imax);
case4_Rho_total = zeros(1, imax);
case4_massflow = zeros(1, imax);

%% ========================================================================
%  SOLVE FOR EACH CASE USING NUMERICAL INTEGRATION
% ========================================================================
```

```
% ------------------------------------------------------------------------
% CASE 1: Area change only (Isentropic flow, no friction, no heat)
% ------------------------------------------------------------------------
fprintf('Solving Case 1: Area change only (isentropic flow)...\n');
```

```
Solving Case 1: Area change only (isentropic flow)...
```

```
% Reset heat and friction for this case
Q_case1 = 0;
f_case1 = 0;
dQ_case1 = 0;                    % No heat addition per step
dQ_over_Cp_TT_case1 = 0;         % No dimensionless heat addition

% Set inlet conditions
Ma_current = Ma_inlet;
TT_current = TT_inlet;
PT_current = PT_inlet;

% Calculate inlet static conditions from total conditions
RT_current = PT_current / (Rgas * TT_current);          % Total density
TS_current = TT_current / (1 + 0.5 * (gama - 1) * Ma_current^2);  % Static temp
PS_current = PT_current / ((1 + 0.5 * (gama - 1) * Ma_current^2)^(gama/(gama-1))); % Static
pressure
RS_current = PS_current / (Rgas * TS_current);          % Static density
SS_current = sqrt(gama * Rgas * TS_current);            % Speed of sound
Ve_current = Ma_current * SS_current;                   % Velocity
massflow_current = RS_current * Ve_current * area(1);   % Mass flow rate

% Store inlet conditions
case1_Ma(1) = Ma_current;
case1_T_total(1) = TT_current;
case1_P_total(1) = PT_current;
case1_Rho_total(1) = RT_current;
case1_T_static(1) = TS_current;
case1_P_static(1) = PS_current;
case1_Rho_static(1) = RS_current;
case1_SoundSpeed(1) = SS_current;
case1_Velocity(1) = Ve_current;
case1_massflow(1) = massflow_current;

% March along the nozzle using forward Euler integration
for i = 1:imax-1
    M = Ma_current;
    V = Ve_current;
    Ro = RS_current;
    T = TS_current;
    P = PS_current;

    % Geometry at current point
    A_current = area(i);
```

```
    D_current = diameter(i);
    dA = area(i+1) - area(i);           % Change in area over this step

    % Helper variable: XM = 1 + (gama-1)*M^2/2
    XM = 1 + 0.5 * (gama - 1) * M^2;

    % Check if we are at sonic condition (M = 1) - singularity
    if abs(M^2 - 1) < 1e-8
        fprintf('Warning: Mach number approaches 1 at i = %d\n', i);
        break;
    end

    % Calculate change in Mach number using the differential equation
    % dM/M = [ (1+gama*M^2)*XM/(2*(1-M^2)) ] * (dq/(Cp*T0))
    %       + [ gama*XM*M^2/(2*(1-M^2)) ] * (f*dx/D)
    %       - [ (2+(gama-1)*M^2)/(2*(1-M^2)) ] * (dA/A)
    %
    % Note: For Case 1, dq = 0 and f = 0, so only the area change term remains

    term_heat = ((1 + gama * M^2) * XM / (2 * (1 - M^2))) * dQ_over_Cp_TT_case1;
    term_friction = (gama * XM * M^2 / (2 * (1 - M^2))) * (f_case1 * dx / D_current);
    term_area = -((2 + (gama - 1) * M^2) / (2 * (1 - M^2))) * (dA / A_current);

    dM_over_M = term_heat + term_friction + term_area;
    dM = dM_over_M * M;

    % Calculate change in velocity
    % dV/V = 0.5*(dq/(Cp*T0)) + (1/XM)*dM/M
    dV_over_V = 0.5 * dQ_over_Cp_TT_case1 + (1 / XM) * dM_over_M;
    dV = dV_over_V * V;

    % Calculate change in density from continuity equation
    % dRo/Ro = -dV/V - dA/A
    dRo_over_Ro = -dV_over_V - (dA / A_current);
    dRo = dRo_over_Ro * Ro;

    % Calculate change in temperature from energy equation
    % dT/T = XM*(dq/(Cp*T0)) - (gama-1)*M^2*dV/V
    dT_over_T = XM * dQ_over_Cp_TT_case1 - (gama - 1) * M^2 * dV_over_V;
    dT = dT_over_T * T;

    % Calculate change in pressure from momentum equation
    % dP/P = -0.5*gama*M^2*(f*dx/D) - gama*M^2*dV/V
    dP_over_P = -0.5 * gama * M^2 * (f_case1 * dx / D_current) - gama * M^2 * dV_over_V;
    dP = dP_over_P * P;

    % Update variables for next step
    Ma_current = M + dM;
    Ve_current = V + dV;
    RS_current = Ro + dRo;
    TS_current = T + dT;
    PS_current = P + dP;
```

```
    % Calculate derived quantities at the new point
    SS_current = sqrt(gama * Rgas * TS_current);                    % Speed of sound
    TT_current = TS_current * (1 + 0.5 * (gama - 1) * Ma_current^2); % Total temperature
    PT_current = PS_current * ((1 + 0.5 * (gama - 1) * Ma_current^2)^(gama/(gama-1))); % Total
pressure
    RT_current = PT_current / (Rgas * TT_current);                  % Total density
    massflow_current = RS_current * Ve_current * area(i+1);         % Mass flow rate

    % Store results
    case1_Ma(i+1) = Ma_current;
    case1_T_total(i+1) = TT_current;
    case1_P_total(i+1) = PT_current;
    case1_Rho_total(i+1) = RT_current;
    case1_T_static(i+1) = TS_current;
    case1_P_static(i+1) = PS_current;
    case1_Rho_static(i+1) = RS_current;
    case1_SoundSpeed(i+1) = SS_current;
    case1_Velocity(i+1) = Ve_current;
    case1_massflow(i+1) = massflow_current;
end

% -------------------------------------------------------------------------
% CASE 2: Area change with heat addition only (no friction)
% -------------------------------------------------------------------------
fprintf('Solving Case 2: Area change with heat addition...\n');
```

```
Solving Case 2: Area change with heat addition...
```

```
% Reset for this case
Q_case2 = Q_total;              % Total heat added [J/kg]
f_case2 = 0;                    % No friction
dQ_case2 = dQ;                  % Heat per step
dQ_over_Cp_TT_case2 = dQ / (Cp * TT_inlet);  % Dimensionless heat per step

% Reset inlet conditions
Ma_current = Ma_inlet;
TT_current = TT_inlet;
PT_current = PT_inlet;

% Calculate inlet static conditions
RT_current = PT_current / (Rgas * TT_current);
TS_current = TT_current / (1 + 0.5 * (gama - 1) * Ma_current^2);
PS_current = PT_current / ((1 + 0.5 * (gama - 1) * Ma_current^2)^(gama/(gama-1)));
RS_current = PS_current / (Rgas * TS_current);
SS_current = sqrt(gama * Rgas * TS_current);
Ve_current = Ma_current * SS_current;
massflow_current = RS_current * Ve_current * area(1);

% Store inlet conditions
case2_Ma(1) = Ma_current;
```

```
case2_T_total(1) = TT_current;
case2_P_total(1) = PT_current;
case2_Rho_total(1) = RT_current;
case2_T_static(1) = TS_current;
case2_P_static(1) = PS_current;
case2_Rho_static(1) = RS_current;
case2_SoundSpeed(1) = SS_current;
case2_Velocity(1) = Ve_current;
case2_massflow(1) = massflow_current;

% March along the nozzle
for i = 1:imax-1
    M = Ma_current;
    V = Ve_current;
    Ro = RS_current;
    T = TS_current;
    P = PS_current;

    A_current = area(i);
    D_current = diameter(i);
    dA = area(i+1) - area(i);
    XM = 1 + 0.5 * (gama - 1) * M^2;

    if abs(M^2 - 1) < 1e-8
        fprintf('Warning: Mach number approaches 1 at i = %d\n', i);
        break;
    end

    % Calculate changes including heat addition
    term_heat = ((1 + gama * M^2) * XM / (2 * (1 - M^2))) * dQ_over_Cp_TT_case2;
    term_friction = (gama * XM * M^2 / (2 * (1 - M^2))) * (f_case2 * dx / D_current);
    term_area = -((2 + (gama - 1) * M^2) / (2 * (1 - M^2))) * (dA / A_current);

    dM_over_M = term_heat + term_friction + term_area;
    dM = dM_over_M * M;

    dV_over_V = 0.5 * dQ_over_Cp_TT_case2 + (1 / XM) * dM_over_M;
    dV = dV_over_V * V;

    dRo_over_Ro = -dV_over_V - (dA / A_current);
    dRo = dRo_over_Ro * Ro;

    dT_over_T = XM * dQ_over_Cp_TT_case2 - (gama - 1) * M^2 * dV_over_V;
    dT = dT_over_T * T;

    dP_over_P = -0.5 * gama * M^2 * (f_case2 * dx / D_current) - gama * M^2 * dV_over_V;
    dP = dP_over_P * P;

    % Update variables
    Ma_current = M + dM;
    Ve_current = V + dV;
    RS_current = Ro + dRo;
```

```
    TS_current = T + dT;
    PS_current = P + dP;

    SS_current = sqrt(gama * Rgas * TS_current);
    TT_current = TS_current * (1 + 0.5 * (gama - 1) * Ma_current^2);
    PT_current = PS_current * ((1 + 0.5 * (gama - 1) * Ma_current^2)^(gama/(gama-1)));
    RT_current = PT_current / (Rgas * TT_current);
    massflow_current = RS_current * Ve_current * area(i+1);

    case2_Ma(i+1) = Ma_current;
    case2_T_total(i+1) = TT_current;
    case2_P_total(i+1) = PT_current;
    case2_Rho_total(i+1) = RT_current;
    case2_T_static(i+1) = TS_current;
    case2_P_static(i+1) = PS_current;
    case2_Rho_static(i+1) = RS_current;
    case2_SoundSpeed(i+1) = SS_current;
    case2_Velocity(i+1) = Ve_current;
    case2_massflow(i+1) = massflow_current;
end

% ------------------------------------------------------------------------
% CASE 3: Area change with friction only (no heat)
% ------------------------------------------------------------------------
fprintf('Solving Case 3: Area change with friction...\n');
```

```
Solving Case 3: Area change with friction...
```

```
% Reset for this case
Q_case3 = 0;                    % No heat addition
f_case3 = friction_factor;      % With friction
dQ_case3 = 0;
dQ_over_Cp_TT_case3 = 0;

% Reset inlet conditions
Ma_current = Ma_inlet;
TT_current = TT_inlet;
PT_current = PT_inlet;

% Calculate inlet static conditions
RT_current = PT_current / (Rgas * TT_current);
TS_current = TT_current / (1 + 0.5 * (gama - 1) * Ma_current^2);
PS_current = PT_current / ((1 + 0.5 * (gama - 1) * Ma_current^2)^(gama/(gama-1)));
RS_current = PS_current / (Rgas * TS_current);
SS_current = sqrt(gama * Rgas * TS_current);
Ve_current = Ma_current * SS_current;
massflow_current = RS_current * Ve_current * area(1);

% Store inlet conditions
case3_Ma(1) = Ma_current;
case3_T_total(1) = TT_current;
```

```
case3_P_total(1) = PT_current;
case3_Rho_total(1) = RT_current;
case3_T_static(1) = TS_current;
case3_P_static(1) = PS_current;
case3_Rho_static(1) = RS_current;
case3_SoundSpeed(1) = SS_current;
case3_Velocity(1) = Ve_current;
case3_massflow(1) = massflow_current;

% March along the nozzle
for i = 1:imax-1
    M = Ma_current;
    V = Ve_current;
    Ro = RS_current;
    T = TS_current;
    P = PS_current;

    A_current = area(i);
    D_current = diameter(i);
    dA = area(i+1) - area(i);
    XM = 1 + 0.5 * (gama - 1) * M^2;

    if abs(M^2 - 1) < 1e-8
        fprintf('Warning: Mach number approaches 1 at i = %d\n', i);
        break;
    end

    % Calculate changes including friction
    term_heat = ((1 + gama * M^2) * XM / (2 * (1 - M^2))) * dQ_over_Cp_TT_case3;
    term_friction = (gama * XM * M^2 / (2 * (1 - M^2))) * (f_case3 * dx / D_current);
    term_area = -((2 + (gama - 1) * M^2) / (2 * (1 - M^2))) * (dA / A_current);

    dM_over_M = term_heat + term_friction + term_area;
    dM = dM_over_M * M;

    dV_over_V = 0.5 * dQ_over_Cp_TT_case3 + (1 / XM) * dM_over_M;
    dV = dV_over_V * V;

    dRo_over_Ro = -dV_over_V - (dA / A_current);
    dRo = dRo_over_Ro * Ro;

    dT_over_T = XM * dQ_over_Cp_TT_case3 - (gama - 1) * M^2 * dV_over_V;
    dT = dT_over_T * T;

    dP_over_P = -0.5 * gama * M^2 * (f_case3 * dx / D_current) - gama * M^2 * dV_over_V;
    dP = dP_over_P * P;

    % Update variables
    Ma_current = M + dM;
    Ve_current = V + dV;
    RS_current = Ro + dRo;
    TS_current = T + dT;
```

```
    PS_current = P + dP;

    SS_current = sqrt(gama * Rgas * TS_current);
    TT_current = TS_current * (1 + 0.5 * (gama - 1) * Ma_current^2);
    PT_current = PS_current * ((1 + 0.5 * (gama - 1) * Ma_current^2)^(gama/(gama-1)));
    RT_current = PT_current / (Rgas * TT_current);
    massflow_current = RS_current * Ve_current * area(i+1);

    case3_Ma(i+1) = Ma_current;
    case3_T_total(i+1) = TT_current;
    case3_P_total(i+1) = PT_current;
    case3_Rho_total(i+1) = RT_current;
    case3_T_static(i+1) = TS_current;
    case3_P_static(i+1) = PS_current;
    case3_Rho_static(i+1) = RS_current;
    case3_SoundSpeed(i+1) = SS_current;
    case3_Velocity(i+1) = Ve_current;
    case3_massflow(i+1) = massflow_current;
end

% -------------------------------------------------------------------------
% CASE 4: Area change with both heat addition and friction
% -------------------------------------------------------------------------
fprintf('Solving Case 4: Area change with heat addition and friction...\n');
```

```
Solving Case 4: Area change with heat addition and friction...
```

```
% Reset for this case
Q_case4 = Q_total;              % Total heat added
f_case4 = friction_factor;      % With friction
dQ_case4 = dQ;
dQ_over_Cp_TT_case4 = dQ / (Cp * TT_inlet);

% Reset inlet conditions
Ma_current = Ma_inlet;
TT_current = TT_inlet;
PT_current = PT_inlet;

% Calculate inlet static conditions
RT_current = PT_current / (Rgas * TT_current);
TS_current = TT_current / (1 + 0.5 * (gama - 1) * Ma_current^2);
PS_current = PT_current / ((1 + 0.5 * (gama - 1) * Ma_current^2)^(gama/(gama-1)));
RS_current = PS_current / (Rgas * TS_current);
SS_current = sqrt(gama * Rgas * TS_current);
Ve_current = Ma_current * SS_current;
massflow_current = RS_current * Ve_current * area(1);

% Store inlet conditions
case4_Ma(1) = Ma_current;
case4_T_total(1) = TT_current;
case4_P_total(1) = PT_current;
```

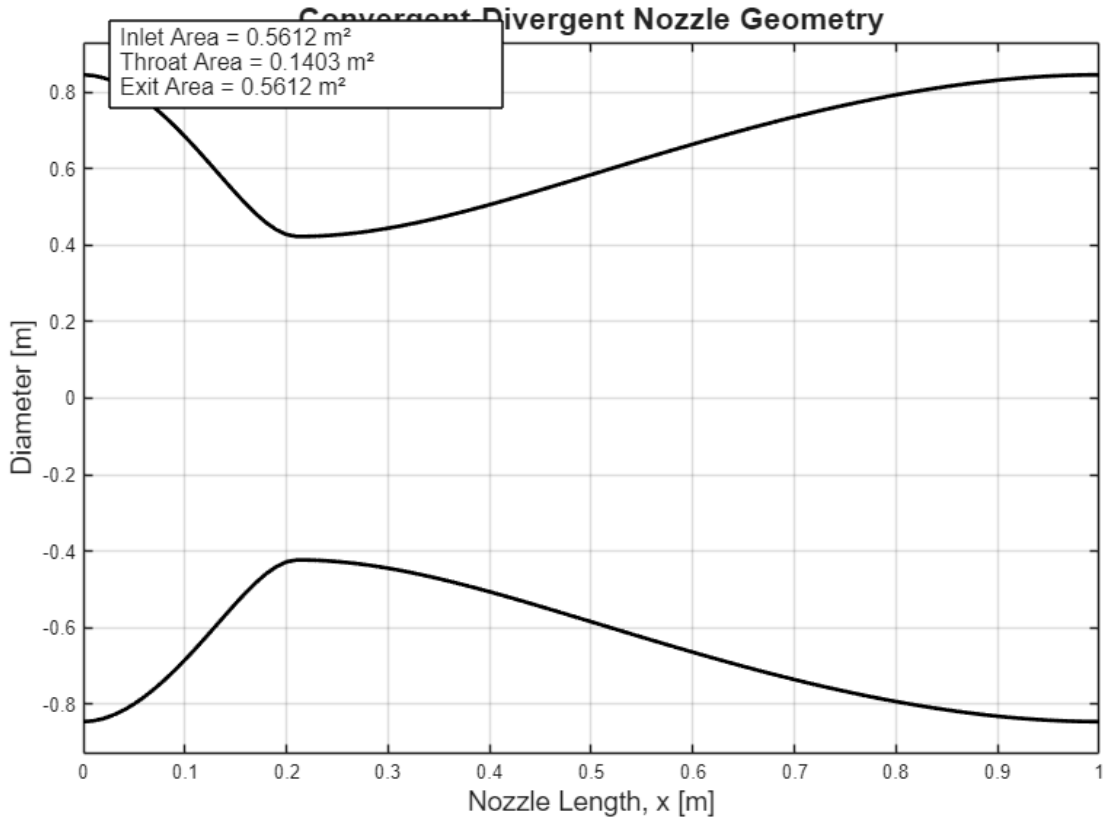
```
case4_Rho_total(1) = RT_current;
case4_T_static(1) = TS_current;
case4_P_static(1) = PS_current;
case4_Rho_static(1) = RS_current;
case4_SoundSpeed(1) = SS_current;
case4_Velocity(1) = Ve_current;
case4_massflow(1) = massflow_current;

% March along the nozzle
for i = 1:imax-1
    M = Ma_current;
    V = Ve_current;
    Ro = RS_current;
    T = TS_current;
    P = PS_current;

    A_current = area(i);
    D_current = diameter(i);
    dA = area(i+1) - area(i);
    XM = 1 + 0.5 * (gama - 1) * M^2;

    if abs(M^2 - 1) < 1e-8
        fprintf('Warning: Mach number approaches 1 at i = %d\n', i);
        break;
    end

    % Calculate changes including both heat and friction
    term_heat = ((1 + gama * M^2) * XM / (2 * (1 - M^2))) * dQ_over_Cp_TT_case4;
    term_friction = (gama * XM * M^2 / (2 * (1 - M^2))) * (f_case4 * dx / D_current);
    term_area = -((2 + (gama - 1) * M^2) / (2 * (1 - M^2))) * (dA / A_current);

    dM_over_M = term_heat + term_friction + term_area;
    dM = dM_over_M * M;

    dV_over_V = 0.5 * dQ_over_Cp_TT_case4 + (1 / XM) * dM_over_M;
    dV = dV_over_V * V;

    dRo_over_Ro = -dV_over_V - (dA / A_current);
    dRo = dRo_over_Ro * Ro;

    dT_over_T = XM * dQ_over_Cp_TT_case4 - (gama - 1) * M^2 * dV_over_V;
    dT = dT_over_T * T;

    dP_over_P = -0.5 * gama * M^2 * (f_case4 * dx / D_current) - gama * M^2 * dV_over_V;
    dP = dP_over_P * P;

    % Update variables
    Ma_current = M + dM;
    Ve_current = V + dV;
    RS_current = Ro + dRo;
    TS_current = T + dT;
    PS_current = P + dP;
```

```
    SS_current = sqrt(gama * Rgas * TS_current);
    TT_current = TS_current * (1 + 0.5 * (gama - 1) * Ma_current^2);
    PT_current = PS_current * ((1 + 0.5 * (gama - 1) * Ma_current^2)^(gama/(gama-1)));
    RT_current = PT_current / (Rgas * TT_current);
    massflow_current = RS_current * Ve_current * area(i+1);

    case4_Ma(i+1) = Ma_current;
    case4_T_total(i+1) = TT_current;
    case4_P_total(i+1) = PT_current;
    case4_Rho_total(i+1) = RT_current;
    case4_T_static(i+1) = TS_current;
    case4_P_static(i+1) = PS_current;
    case4_Rho_static(i+1) = RS_current;
    case4_SoundSpeed(i+1) = SS_current;
    case4_Velocity(i+1) = Ve_current;
    case4_massflow(i+1) = massflow_current;
end

fprintf('All cases solved successfully!\n');
```

```
All cases solved successfully!
```

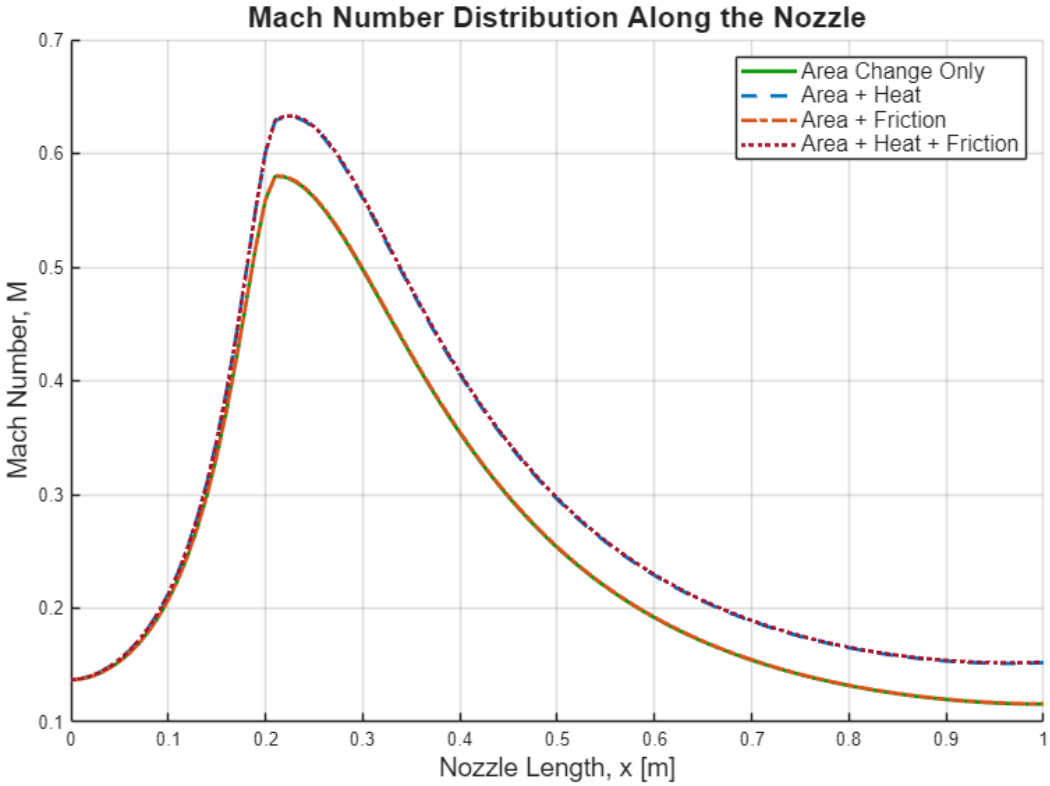
```
%% ========================================================================
%  PLOT THE RESULTS
% ========================================================================

% Set up plotting parameters
linewidth_val = 2;
fontsize_val = 14;
marker_size = 6;

% Color definitions
color_area_only = [0 0.6 0];        % Dark green
color_heat_only = [0 0.447 0.741];  % Blue
color_friction_only = [0.85 0.325 0.098]; % Orange
color_heat_friction = [0.635 0.078 0.184]; % Dark red

% Line styles
style_area_only = '-';
style_heat_only = '--';
style_friction_only = '-.';
style_heat_friction = ':';

% Legend labels
legend_labels = {'Area Change Only', 'Area + Heat', 'Area + Friction', 'Area + Heat +
Friction'};

% ------------------------------------------------------------------------
% FIGURE 1: Nozzle Geometry (Diameter vs. Length)
% ------------------------------------------------------------------------
```

```
figure('Name', 'Nozzle Geometry', 'Position', [100, 100, 900, 600]);
plot(x_coordinate, diameter, 'k-', 'LineWidth', linewidth_val);
hold on;
plot(x_coordinate, -diameter, 'k-', 'LineWidth', linewidth_val);
grid on;
xlabel('Nozzle Length, x [m]', 'FontSize', fontsize_val);
ylabel('Diameter [m]', 'FontSize', fontsize_val);
title('Convergent-Divergent Nozzle Geometry', 'FontSize', fontsize_val+2);

xlim([0, XL]);
ylim([-max(diameter)*1.1, max(diameter)*1.1]);
annotation('textbox', [0.15, 0.85, 0.3, 0.1], ...
    'String', sprintf('Inlet Area = %.4f m²\nThroat Area = %.4f m²\nExit Area = %.4f m²', ...
    area(1), min(area), area(end)), ...
    'FontSize', 12, 'BackgroundColor', 'white', 'EdgeColor', 'black');
```



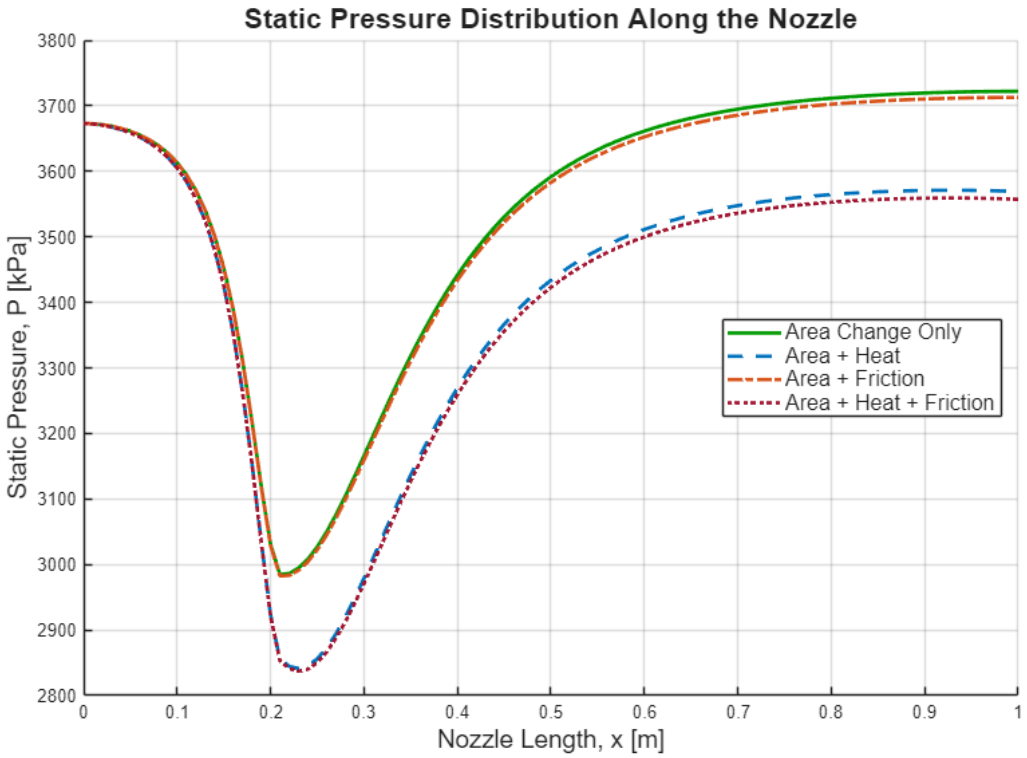

```
% ------------------------------------------------------------------------
% FIGURE 2: Mach Number vs. Nozzle Length
% ------------------------------------------------------------------------
figure('Name', 'Mach Number Distribution', 'Position', [100, 100, 900, 600]);
hold on;
plot(x_coordinate, case1_Ma, style_area_only, 'Color', color_area_only, 'LineWidth',
linewidth_val);
plot(x_coordinate, case2_Ma, style_heat_only, 'Color', color_heat_only, 'LineWidth',
linewidth_val);
plot(x_coordinate, case3_Ma, style_friction_only, 'Color', color_friction_only, 'LineWidth',
linewidth_val);
```

```
 plot(x_coordinate, case4_Ma, style_heat_friction, 'Color', color_heat_friction, 'LineWidth',
linewidth_val);
 grid on;
 xlabel('Nozzle Length, x [m]', 'FontSize', fontsize_val);
 ylabel('Mach Number, M', 'FontSize', fontsize_val);
 title('Mach Number Distribution Along the Nozzle', 'FontSize', fontsize_val+2);
 legend(legend_labels, 'Location', 'best', 'FontSize', fontsize_val-2);
 hold off;
```

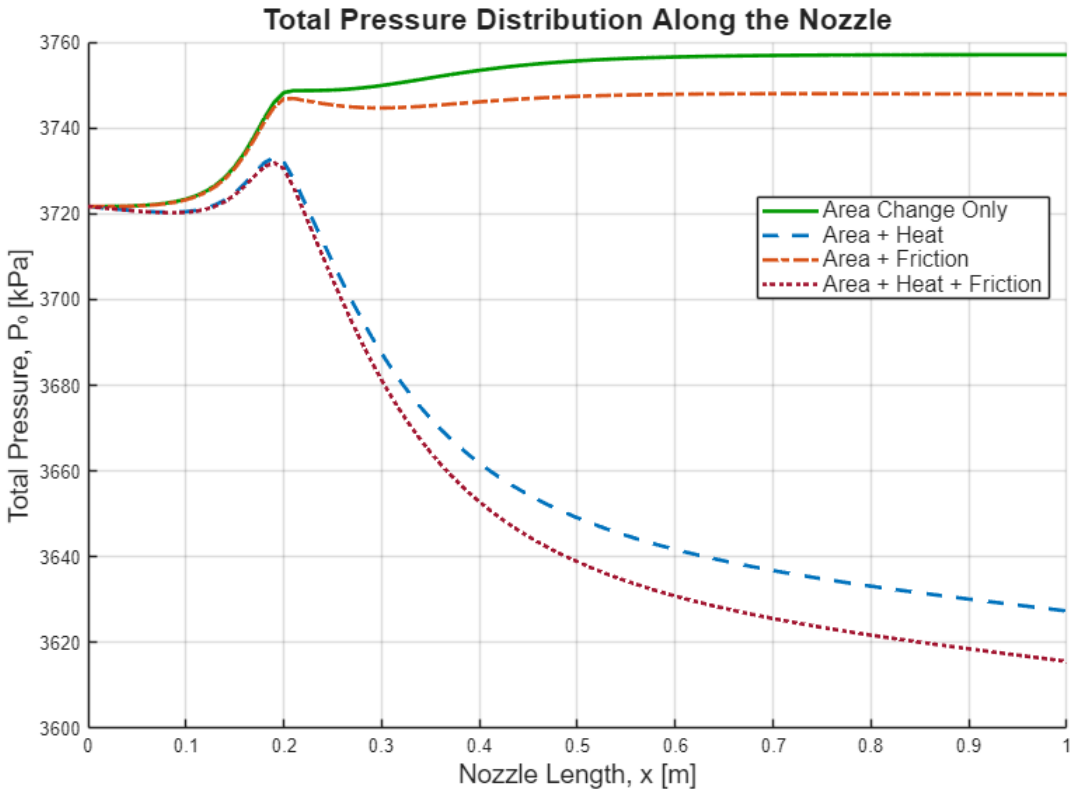



```
 % ------------------------------------------------------------------------
 % FIGURE 3: Static Pressure vs. Nozzle Length
 % ------------------------------------------------------------------------
 figure('Name', 'Static Pressure Distribution', 'Position', [100, 100, 900, 600]);
 hold on;
 plot(x_coordinate, case1_P_static/1e3, style_area_only, 'Color', color_area_only, 'LineWidth',
linewidth_val);
 plot(x_coordinate, case2_P_static/1e3, style_heat_only, 'Color', color_heat_only, 'LineWidth',
linewidth_val);
 plot(x_coordinate, case3_P_static/1e3, style_friction_only, 'Color', color_friction_only,
'LineWidth', linewidth_val);
 plot(x_coordinate, case4_P_static/1e3, style_heat_friction, 'Color', color_heat_friction,
'LineWidth', linewidth_val);
 grid on;
 xlabel('Nozzle Length, x [m]', 'FontSize', fontsize_val);
 ylabel('Static Pressure, P [kPa]', 'FontSize', fontsize_val);
 title('Static Pressure Distribution Along the Nozzle', 'FontSize', fontsize_val+2);
 legend(legend_labels, 'Location', 'best', 'FontSize', fontsize_val-2);
```

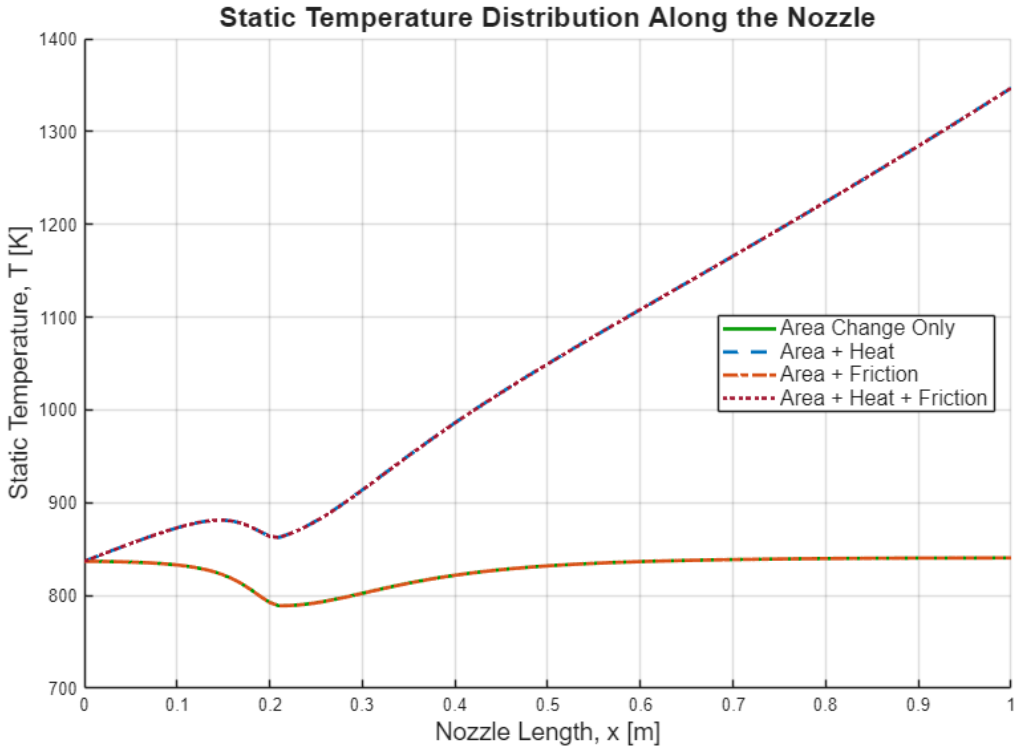
```
hold off;
```



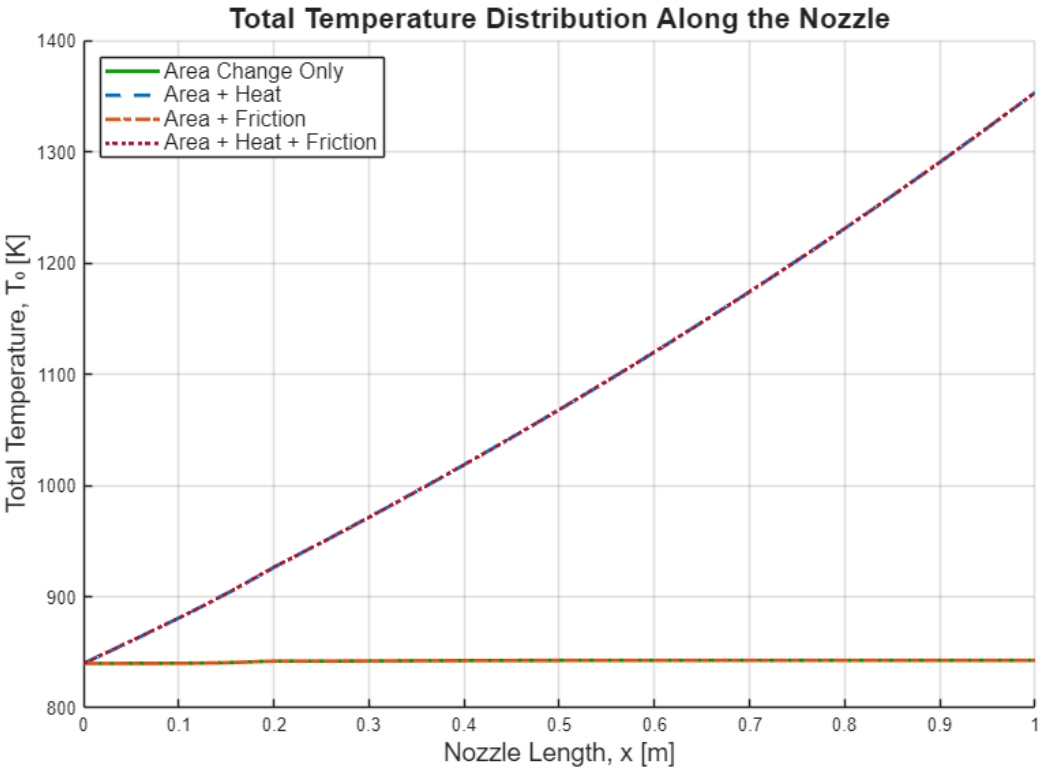

```
% ------------------------------------------------------------------------
% FIGURE 4: Total Pressure vs. Nozzle Length
% ------------------------------------------------------------------------
figure('Name', 'Total Pressure Distribution', 'Position', [100, 100, 900, 600]);
hold on;
plot(x_coordinate, case1_P_total/1e3, style_area_only, 'Color', color_area_only, 'LineWidth',
linewidth_val);
plot(x_coordinate, case2_P_total/1e3, style_heat_only, 'Color', color_heat_only, 'LineWidth',
linewidth_val);
plot(x_coordinate, case3_P_total/1e3, style_friction_only, 'Color', color_friction_only,
'LineWidth', linewidth_val);
plot(x_coordinate, case4_P_total/1e3, style_heat_friction, 'Color', color_heat_friction,
'LineWidth', linewidth_val);
grid on;
xlabel('Nozzle Length, x [m]', 'FontSize', fontsize_val);
ylabel('Total Pressure, P₀ [kPa]', 'FontSize', fontsize_val);
title('Total Pressure Distribution Along the Nozzle', 'FontSize', fontsize_val+2);
legend(legend_labels, 'Location', 'best', 'FontSize', fontsize_val-2);
hold off;
```



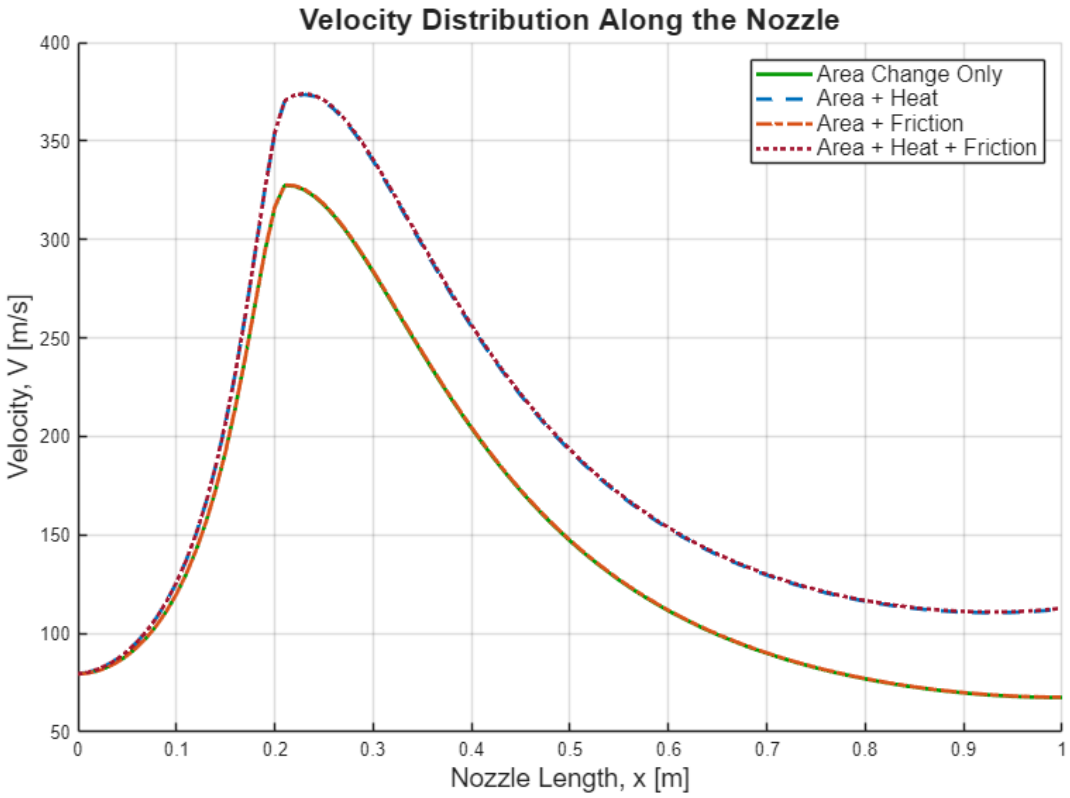

```
% ------------------------------------------------------------------------
% FIGURE 5: Static Temperature vs. Nozzle Length
% ------------------------------------------------------------------------
figure('Name', 'Static Temperature Distribution', 'Position', [100, 100, 900, 600]);
hold on;
plot(x_coordinate, case1_T_static, style_area_only, 'Color', color_area_only, 'LineWidth',
linewidth_val);
plot(x_coordinate, case2_T_static, style_heat_only, 'Color', color_heat_only, 'LineWidth',
linewidth_val);
plot(x_coordinate, case3_T_static, style_friction_only, 'Color', color_friction_only,
'LineWidth', linewidth_val);
plot(x_coordinate, case4_T_static, style_heat_friction, 'Color', color_heat_friction,
'LineWidth', linewidth_val);
grid on;
xlabel('Nozzle Length, x [m]', 'FontSize', fontsize_val);
ylabel('Static Temperature, T [K]', 'FontSize', fontsize_val);
title('Static Temperature Distribution Along the Nozzle', 'FontSize', fontsize_val+2);
legend(legend_labels, 'Location', 'best', 'FontSize', fontsize_val-2);
hold off;
```



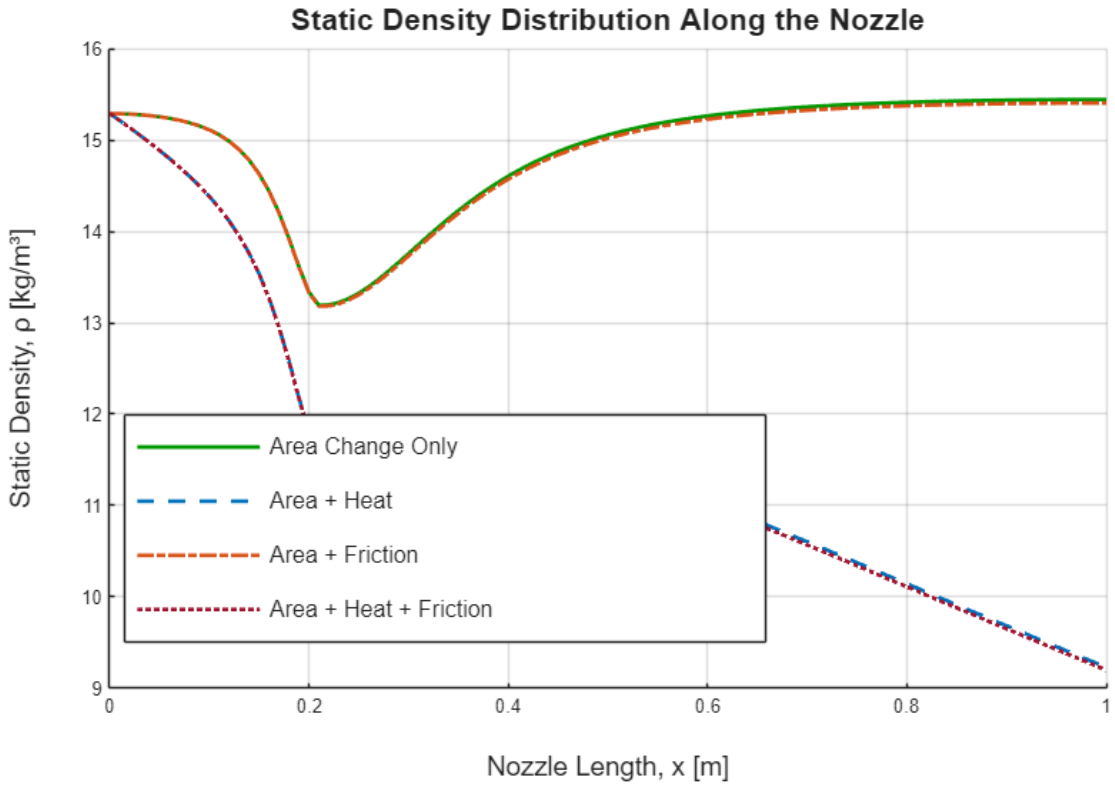

```
% ------------------------------------------------------------------------
% FIGURE 6: Total Temperature vs. Nozzle Length
% ------------------------------------------------------------------------
figure('Name', 'Total Temperature Distribution', 'Position', [100, 100, 900, 600]);
hold on;
plot(x_coordinate, case1_T_total, style_area_only, 'Color', color_area_only, 'LineWidth',
linewidth_val);
plot(x_coordinate, case2_T_total, style_heat_only, 'Color', color_heat_only, 'LineWidth',
linewidth_val);
plot(x_coordinate, case3_T_total, style_friction_only, 'Color', color_friction_only,
'LineWidth', linewidth_val);
plot(x_coordinate, case4_T_total, style_heat_friction, 'Color', color_heat_friction,
'LineWidth', linewidth_val);
grid on;
xlabel('Nozzle Length, x [m]', 'FontSize', fontsize_val);
ylabel('Total Temperature, T₀ [K]', 'FontSize', fontsize_val);
title('Total Temperature Distribution Along the Nozzle', 'FontSize', fontsize_val+2);
legend(legend_labels, 'Location', 'best', 'FontSize', fontsize_val-2);
hold off;
```



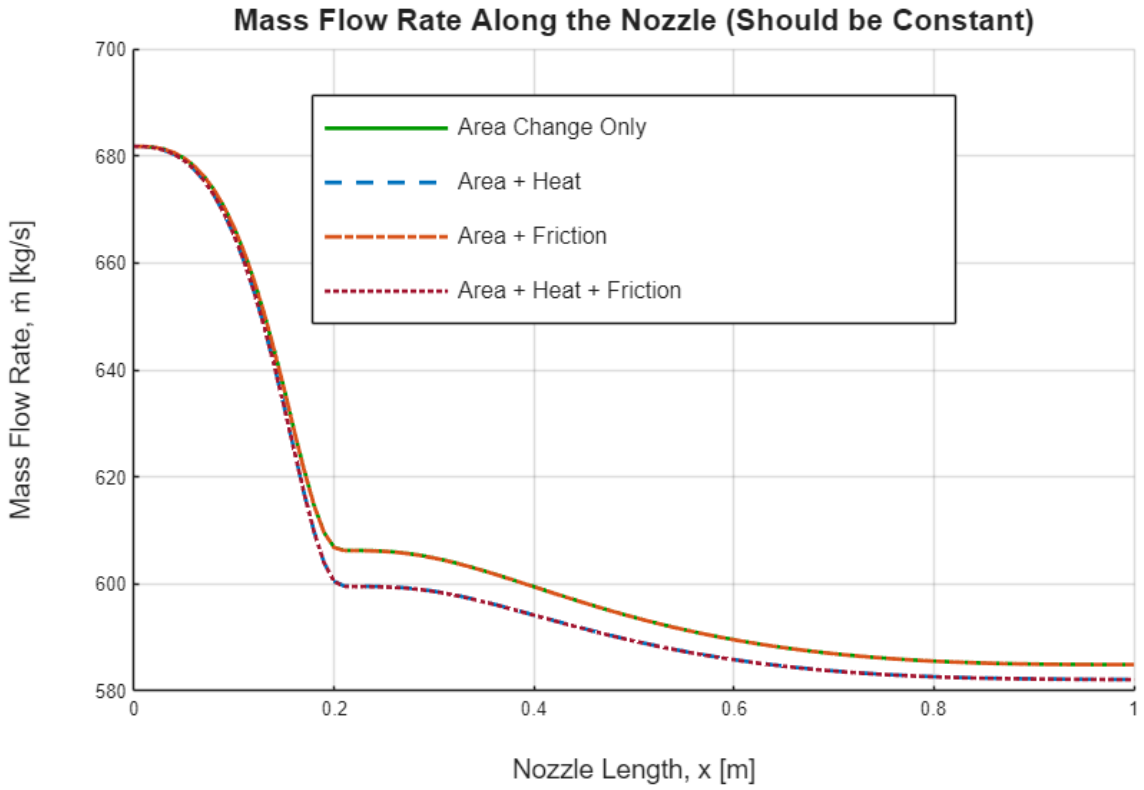

```
% ------------------------------------------------------------------------
% FIGURE 7: Velocity vs. Nozzle Length
% ------------------------------------------------------------------------
figure('Name', 'Velocity Distribution', 'Position', [100, 100, 900, 600]);
hold on;
plot(x_coordinate, case1_Velocity, style_area_only, 'Color', color_area_only, 'LineWidth',
linewidth_val);
plot(x_coordinate, case2_Velocity, style_heat_only, 'Color', color_heat_only, 'LineWidth',
linewidth_val);
plot(x_coordinate, case3_Velocity, style_friction_only, 'Color', color_friction_only,
'LineWidth', linewidth_val);
plot(x_coordinate, case4_Velocity, style_heat_friction, 'Color', color_heat_friction,
'LineWidth', linewidth_val);
grid on;
xlabel('Nozzle Length, x [m]', 'FontSize', fontsize_val);
ylabel('Velocity, V [m/s]', 'FontSize', fontsize_val);
title('Velocity Distribution Along the Nozzle', 'FontSize', fontsize_val+2);
legend(legend_labels, 'Location', 'best', 'FontSize', fontsize_val-2);
hold off;
```




```
% ------------------------------------------------------------------------
% FIGURE 8: Static Density vs. Nozzle Length
% ------------------------------------------------------------------------
figure('Name', 'Static Density Distribution', 'Position', [100, 100, 900, 600]);
hold on;
plot(x_coordinate, case1_Rho_static, style_area_only, 'Color', color_area_only, 'LineWidth',
linewidth_val);
plot(x_coordinate, case2_Rho_static, style_heat_only, 'Color', color_heat_only, 'LineWidth',
linewidth_val);
plot(x_coordinate, case3_Rho_static, style_friction_only, 'Color', color_friction_only,
'LineWidth', linewidth_val);
plot(x_coordinate, case4_Rho_static, style_heat_friction, 'Color', color_heat_friction,
'LineWidth', linewidth_val);
grid on;
xlabel('Nozzle Length, x [m]', 'FontSize', fontsize_val);
ylabel('Static Density, ρ [kg/m³]', 'FontSize', fontsize_val);
title('Static Density Distribution Along the Nozzle', 'FontSize', fontsize_val+2);
legend(legend_labels, 'Location', 'best', 'FontSize', fontsize_val-2);
hold off;
```




```
% ------------------------------------------------------------------------
% FIGURE 9: Mass Flow Rate vs. Nozzle Length (should be constant for steady flow)
% ------------------------------------------------------------------------
figure('Name', 'Mass Flow Rate', 'Position', [100, 100, 900, 600]);
hold on;
plot(x_coordinate, case1_massflow, style_area_only, 'Color', color_area_only, 'LineWidth',
linewidth_val);
plot(x_coordinate, case2_massflow, style_heat_only, 'Color', color_heat_only, 'LineWidth',
linewidth_val);
plot(x_coordinate, case3_massflow, style_friction_only, 'Color', color_friction_only,
'LineWidth', linewidth_val);
plot(x_coordinate, case4_massflow, style_heat_friction, 'Color', color_heat_friction,
'LineWidth', linewidth_val);
grid on;
xlabel('Nozzle Length, x [m]', 'FontSize', fontsize_val);
ylabel('Mass Flow Rate, ṁ [kg/s]', 'FontSize', fontsize_val);
title('Mass Flow Rate Along the Nozzle (Should be Constant)', 'FontSize', fontsize_val+2);
legend(legend_labels, 'Location', 'best', 'FontSize', fontsize_val-2);
hold off;
```




```
% ------------------------------------------------------------------------
% Print summary of results at the exit
% ------------------------------------------------------------------------
fprintf('\n========== RESULTS SUMMARY AT NOZZLE EXIT ==========\n');
```

```
========== RESULTS SUMMARY AT NOZZLE EXIT ==========
```

```
fprintf('Case 1 (Area only):       M = %.4f, P = %.2f kPa, T = %.2f K\n', ...
    case1_Ma(end), case1_P_static(end)/1e3, case1_T_static(end));
```

```
Case 1 (Area only):       M = 0.1157, P = 3722.09 kPa, T = 840.51 K
```

```
fprintf('Case 2 (Area + Heat):     M = %.4f, P = %.2f kPa, T = %.2f K\n', ...
    case2_Ma(end), case2_P_static(end)/1e3, case2_T_static(end));
```

```
Case 2 (Area + Heat):     M = 0.1521, P = 3569.20 kPa, T = 1346.92 K
```

```
fprintf('Case 3 (Area + Friction): M = %.4f, P = %.2f kPa, T = %.2f K\n', ...
    case3_Ma(end), case3_P_static(end)/1e3, case3_T_static(end));
```

```
Case 3 (Area + Friction): M = 0.1160, P = 3712.75 kPa, T = 840.50 K
```

```
fprintf('Case 4 (Area + Heat + Friction): M = %.4f, P = %.2f kPa, T = %.2f K\n', ...
    case4_Ma(end), case4_P_static(end)/1e3, case4_T_static(end));
```

```
Case 4 (Area + Heat + Friction): M = 0.1526, P = 3557.27 kPa, T = 1346.89 K
```

```
fprintf('====================================================\n');
```

```
====================================================
```